When finally you and your best friend memorize Morse code (for that's the only way you can become proficient at sending and receiving it), you can also use it vocally as a substitute for normal speech. For maximum speed, you pronounce a dot as *dih* (or *dit* for the last dot of a letter) and a dash as *dah*. In the same way that Morse code reduces written language to dots and dashes, the spoken version of the code reduces speech to just two vowel sounds.

The key word here is *two*. Two types of blinks, two vowel sounds, two different anything, really, can with suitable combinations convey all types of information.

Chapter Two

Codes and Combinations

Morse code was invented by Samuel Finley Breese Morse (1791–1872), whom we shall meet more properly later in this book. The invention of Morse code goes hand in hand with the invention of the telegraph, which we'll also examine in more detail. Just as Morse code provides a good introduction to the nature of codes, the telegraph provides a good introduction to the hardware of the computer.

Most people find Morse code easier to send than to receive. Even if you don't have Morse code memorized, you can simply use this table, conveniently arranged in alphabetical order:

A	·–	J	·–––	S	···
B	–···	K	–·–	T	–
C	–·–·	L	·–··	U	··–
D	–··	M	––	V	···–
E	·	N	–·	W	·––
F	··–·	O	–––	X	–··–
G	––·	P	·––·	Y	–·––
H	····	Q	––·–	Z	––··
I	··	R	·–·		

Receiving Morse code and translating it back into words is considerably harder and more time consuming than sending because you must work backward to figure out the letter that corresponds to a particular coded sequence of dots and dashes. For example, if you receive a dash-dot-dash-dash, you have to scan through the table letter by letter before you finally discover that the code is the letter Y.

The problem is that we have a table that provides this translation:

Alphabetical letter → Morse code dots and dashes

But we *don't* have a table that lets us go backward:

Morse code dots and dashes → Alphabetical letter

In the early stages of learning Morse code, such a table would certainly be convenient. But it's not at all obvious how we could construct it. There's nothing in those dots and dashes that we can put into alphabetical order.

So let's forget about alphabetical order. Perhaps a better approach to organizing the codes might be to group them depending on how many dots and dashes they have. For example, a Morse code sequence that contains either one dot or one dash can represent only two letters, which are E and T:

·	E
–	T

A combination of exactly two dots or dashes gives us four more letters—I, A, N, and M:

··	I	–·	N
·–	A	––	M

A pattern of three dots or dashes gives us eight more letters:

···	S	–··	D
··–	U	–·–	K
·–·	R	––·	G
·––	W	–––	O

And finally (if we want to stop this exercise before dealing with numbers and punctuation marks), sequences of four dots and dashes give us 16 more characters:

····	H	–···	B
···–	V	–··–	X
··–·	F	–·–·	C
··––	Ü	–·––	Y
·–··	L	––··	Z
·–·–	Ä	––·–	Q
·––·	P	–––·	Ö
·–––	J	————	Ş

Taken together, these four tables contain 2 plus 4 plus 8 plus 16 codes for a total of 30 letters, 4 more than are needed for the 26 letters of the Latin alphabet. For this reason, you'll notice that 4 of the codes in the last table are for accented letters.

These four tables might help you translate with greater ease when someone is sending you Morse code. After you receive a code for a particular letter, you know how many dots and dashes it has, and you can at least go to the right table to look it up. Each table is organized so that you find the all-dots code in the upper left and the all-dashes code in the lower right.

Can you see a pattern in the *size* of the four tables? Notice that each table has twice as many codes as the table before it. This makes sense: Each table has all the codes in the previous table followed by a dot, and all the codes in the previous table followed by a dash.

We can summarize this interesting trend this way:

Number of Dots and Dashes	Number of Codes
1	2
2	4
3	8
4	16

Each of the four tables has twice as many codes as the table before it, so if the first table has 2 codes, the second table has 2 × 2 codes, and the third table has 2 × 2 × 2 codes. Here's another way to show that:

Number of Dots and Dashes	Number of Codes
1	2
2	2×2
3	$2 \times 2 \times 2$
4	$2 \times 2 \times 2 \times 2$

Of course, once we have a number multiplied by itself, we can start using exponents to show powers. For example, $2 \times 2 \times 2 \times 2$ can be written as 2^4 (*2 to the 4th power*). The numbers 2, 4, 8, and 16 are all powers of 2 because you can calculate them by multiplying 2 by itself. So our summary can also be shown like this:

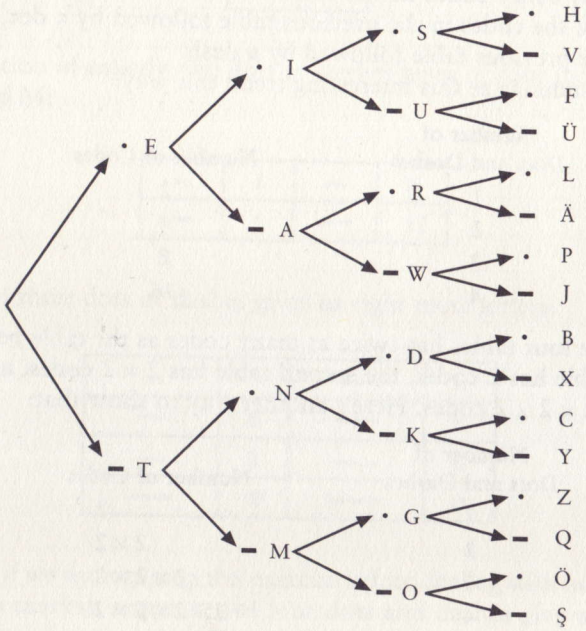
Number of Dots and Dashes	Number of Codes
1	2^1
2	2^2
3	2^3
4	2^4

This table has become very simple. The number of codes is simply 2 to the power of the number of dots and dashes. We might summarize the table data in this simple formula:

$$\text{number of codes} = 2^{\text{number of dots and dashes}}$$

Powers of 2 tend to show up a lot in codes, and we'll see another example in the next chapter.

To make the process of decoding Morse code even easier, we might want to draw something like the big treelike table shown here.



This table shows the letters that result from each particular consecutive sequence of dots and dashes. To decode a particular sequence, follow the arrows from left to right. For example, suppose you want to know which letter corresponds to the code dot-dash-dot. Begin at the left and choose the dot; then continue moving right along the arrows and choose the dash and then another dot. The letter is R, shown next to the last dot.

If you think about it, constructing such a table was probably necessary for defining Morse code in the first place. First, it ensures that you don't make the dumb mistake of using the same code for two different letters! Second, you're assured of using all the possible codes without making the sequences of dots and dashes unnecessarily long.

At the risk of extending this table beyond the limits of the printed page, we could continue it for codes of five dots and dashes and more. A sequence of exactly five dots and dashes gives us 32 ($2\times2\times2\times2\times2$, or 2^5) additional codes. Normally that would be enough for the 10 numbers and the 16 punctuation symbols defined in Morse code, and indeed the numbers are encoded with five dots and dashes. But many of the other codes that use a sequence of five dots and dashes represent accented letters rather than punctuation marks.

To include all the punctuation marks, the system must be expanded to six dots and dashes, which gives us 64 ($2\times2\times2\times2\times2\times2$, or 2^6) additional codes for a grand total of $2+4+8+16+32+64$, or 126, characters. That's overkill for Morse code, which leaves many of these longer codes "undefined." The word *undefined* used in this context refers to a code that doesn't stand for anything. If you were receiving Morse code and you got an undefined code, you could be pretty sure that somebody made a mistake.

Because we were clever enough to develop this little formula,

$$\text{number of codes} = 2^{\text{number of dots and dashes}}$$

we could continue figuring out how many codes we get from using longer sequences of dots and dashes:

Number of Dots and Dashes	Number of Codes
1	$2^1 = 2$
2	$2^2 = 4$
3	$2^3 = 8$
4	$2^4 = 16$
5	$2^5 = 32$
6	$2^6 = 64$
7	$2^7 = 128$
8	$2^8 = 256$
9	$2^9 = 512$
10	$2^{10} = 1024$

Fortunately, we don't have to actually write out all the possible codes to determine how many there would be. All we have to do is multiply 2 by itself over and over again.

Morse code is said to be a *binary* (literally meaning *two by two*) code because the components of the code consist of only two things—a dot and a dash. That's similar to a coin, which can land only on the head side or the tail side. Binary objects (such as coins) and binary codes (such as Morse code) are always described by powers of two.

What we're doing by analyzing binary codes is a simple exercise in the branch of mathematics known as *combinatorics* or *combinatorial analysis*. Traditionally, combinatorial analysis is used most often in the fields of probability and statistics because it involves determining the number of ways that things, like coins and dice, can be combined. But it also helps us understand how codes can be put together and taken apart.

Chapter Three

Braille and Binary Codes

Samuel Morse wasn't the first person to successfully translate the letters of written language to an interpretable code. Nor was he the first person to be remembered more as the name of his code than as himself. That honor must go to a blind French teenager born some 18 years after Samuel Morse but who made his mark much more precociously. Little is known of his life, but what is known makes a compelling story.

Louis Braille was born in 1809 in Coupvray, France, just 25 miles east of Paris. His father was a harness maker. At the age of three—an age when young boys shouldn't be playing in their fathers' workshops—he accidentally stuck a pointed tool in his eye. The wound became infected, and the infection spread to his other eye, leaving him totally blind. Normally he would have been doomed to a life of ignorance and poverty (as most blind people were in those days), but young Louis's intelligence and desire to learn were soon recognized. Through the intervention of the village priest and a schoolteacher, he first attended school in the village with the other children and at the age of 10 was sent to the Royal Institution for Blind Youth in Paris.